

**IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
TYLER DIVISION**

AMERICAN VIDEO GRAPHICS, L.P.,	§	
	§	
Plaintiff,	§	CIVIL ACTION
	§	
vs.	§	6:05cv006 LED
	§	
MICROSOFT CORPORATION,	§	
	§	JURY TRIAL
Defendant.	§	

COMPLAINT AGAINST SEVERED INTERVENOR MICROSOFT CORPORATION

Plaintiff American Video Graphics, L.P. (“AVG”), files this Complaint against severed intervenor Microsoft Corporation (“Microsoft”) for infringement of U.S. Patent Nos. 4,694,286 (“the ’286 Patent”) and 4,761,642 (“the ’642 Patent”) (collectively, “the Patents-in-Suit”) under 35 U.S.C. § 271. Copies of each of the Patents-in-Suit are attached as Exhibits A & B.

BACKGROUND

This action is severed from, and is a continuation of, Civil Action No. 6:04-CV-397-LED, filed August 23, 2004. Microsoft intervened in that matter by way of its Complaint in Intervention filed on December 27, 2004. The Court issued an Order on January 13, 2005 severing all claims and defenses between AVG and Microsoft with regard to the ’642 and ’286 Patents, and ordering that such claims and defenses proceed separately in this action. AVG therefore files this Complaint against Microsoft in accordance with the Court’s Order of January 13, 2005.

PARTIES

1. Plaintiff AVG is a limited partnership, organized under the laws of the State of Texas. AVG maintains its principal place of business at 505 E. Travis Street, Suite 210, Marshall, Texas 75670.

2. Upon information and belief, Defendant Microsoft is, and at all times relevant herein was, a corporation organized under the laws of the State of Delaware, with its principal place of business at One Microsoft Way, Redmond, WA 98052. Microsoft manufactures for sale and/or sells computer software to consumers in the United States and, more particularly, in the Eastern District of Texas.

JURISDICTION AND VENUE

3. This is an action for patent infringement arising under the patent laws of the United States, Title 35, United States Code. This Court has exclusive subject matter jurisdiction over this case for patent infringement under 28 U.S.C. §1338(a).

4. This Court has personal jurisdiction over Microsoft. Microsoft has conducted and does conduct business within the State of Texas. Microsoft, directly or through intermediaries (including distributors, retailers, and others), ships, distributes, offers for sale, sells, and advertises (including the provision of an interactive web page) its products in the United States, the State of Texas, and the Eastern District of Texas. Microsoft has purposefully and voluntarily placed one or more of its infringing products, as described below in Counts 1 & 2, into the stream of commerce with the expectation that they will be purchased by consumers in the Eastern District of Texas. These infringing products have been and continue to be purchased by consumers in the Eastern District of Texas. Microsoft has committed the tort of patent

infringement within the State of Texas and, more particularly, within the Eastern District of Texas.

5. Venue is proper in the Eastern District of Texas under 28 U.S.C. §§ 1391 and 1400(b).

COUNT 1: INFRINGEMENT OF U.S. PATENT NO. 4,694,286

6. AVG refers to and incorporates herein the allegations of Paragraphs 1-5 above.

7. United States Patent No. 4,694,286 (“the ’286 Patent”), entitled “Apparatus and Method for Modifying Displayed Color Images,” was duly and legally issued by the United States Patent and Trademark Office on September 15, 1987, after full and fair examination. AVG is the assignee of all rights, title, and interest in and to the ’286 Patent and possesses all rights of recovery under the ’286 Patent.

8. Microsoft manufactures and/or sells software, such as Windows XP, that includes software for modifying the color of pixels in a color image by enabling a user to (i) select a pixel in the image, (ii) enter new HLS values to specify a new color, and (iii) modify the selected pixel, and all pixels in the image associated with that pixel, to the newly specified color.

9. AVG has at all times complied with 35 U.S.C. § 287.

10. Microsoft has knowledge of the ’286 Patent, and has not ceased its infringing activities. Microsoft’s infringement of the ’286 Patent has been and continues to be willful and deliberate.

11. As a result of Microsoft’s acts of infringement, AVG has suffered and will continue to suffer damages in an amount to be proved at trial.

COUNT 2: INFRINGEMENT OF U.S. PATENT NO. 4,761,642

12. AVG refers to and incorporates herein the allegations of Paragraphs 1-5 above.

13. United States Patent No. 4,761,642 (“the ’642 Patent”), entitled “System for Providing Data Communication Between a Computer Terminal and a Plurality of Concurrent Processes Running on a Multiple Process Computer,” was duly and legally issued by the United States Patent and Trademark Office on August 2, 1988, after full and fair examination. AVG is the assignee of all rights, title, and interest in and to the ’642 Patent and possesses all rights of recovery under the ’642 Patent.

14. Microsoft manufactures and/or sells operating systems, such as Windows XP, Windows 2000, Windows ME, and Windows 98. Microsoft's operating systems, in conjunction with integrated and/or included applications, such as Internet Explorer, display output from and receive user input to multiple concurrent processes using multiple windows.

15. By virtue of the foregoing, Microsoft is infringing one or more claims of the ’642 Patent under 35 U.S.C. § 271 by performing, without authority, one or more of the following acts: (a) making, using, offering to sell, and selling within the United States the invention of one or more claims of the ’642 Patent; (b) importing into the United States the invention of one or more claims of the ’642 Patent; and (c) inducing infringement of one or more claims of the ’642 Patent.

16. AVG has at all times complied with 35 U.S.C. § 287.

17. Microsoft has knowledge of the ’642 Patent, and has not ceased its infringing activities. Microsoft’s infringement of the ’642 Patent has been and continues to be willful and deliberate.

18. As a result of Microsoft’s acts of infringement, AVG has suffered and will continue to suffer damages in an amount to be proved at trial.

PRAYER FOR RELIEF

AVG prays for the following relief:

A. A judgment that Microsoft has infringed the Patents-in-Suit as alleged herein, directly and/or indirectly by way of inducing or contributing to infringement of the Patents-in-Suit, as alleged herein;

B. A judgment and order requiring Microsoft to pay AVG damages under 35 U.S.C. § 284, including treble damages for willful infringement as provided by 35 U.S.C. § 284, and supplemental damages for any continuing post-verdict infringement up until entry of the final Judgment with an accounting as needed;

C. A judgment and order requiring Microsoft to pay AVG pre-judgment and post-judgment interest on the damages awarded;

D. A judgment and order finding this to be an exceptional case and requiring Microsoft to pay the costs of this action (including all disbursements) and attorneys' fees as provided by 35 U.S.C. § 285; and

F. Such other and further relief as the Court deems just and equitable.

DEMAND FOR JURY TRIAL

AVG hereby demands that all issues be determined by jury.

DATED: January 20, 2005

Respectfully submitted,

McKOOL SMITH, P.C.

/s/ Sam Baxter

Sam Baxter
Attorney-in-Charge
Texas State Bar No. 01938000
505 E. Travis, Suite 105
P.O. Box O
Marshall, Texas 75670
sbaxter@mckoolsmith.com
Telephone: (903) 927-2111
Telecopier: (903) 927-2622

Mike McKool, Jr.
Texas State Bar No. 13732100
mmckool@mckoolsmith.com
300 Crescent Court, Suite 1500
Dallas, Texas 75201
Telephone: (214) 978-4000
Telecopier: (214) 978-4044

T. Gordon White
Texas State Bar No. 21333000
gwhite@mckoolsmith.com
300 West Sixth Street, Suite 1700
Austin, Texas 78701
Telephone: (512) 692-8700
Telecopier: (512) 692-8744

**ATTORNEYS FOR PLAINTIFF
AMERICAN VIDEO GRAPHICS, LP.**

CERTIFICATE OF SERVICE

The undersigned certifies that the foregoing document was filed electronically in compliance with Local Rule CV-5(a). As such, this motion was served on all counsel who have consented to electronic service. Local Rule CV-5(a)(3)(A). Pursuant to Fed. R. Civ. P. 5(d) and Local Rule CV-5(e), all other counsel of record not deemed to have consented to electronic service were served with a true and correct copy of the foregoing by U.S. mail, on this the 20th day of January, 2005.

/s/ Sam Baxter
Sam Baxter

United States Patent [19]

Bergstedt

[11] Patent Number: 4,694,286
[45] Date of Patent: Sep. 15, 1987

[54] APPARATUS AND METHOD FOR MODIFYING DISPLAYED COLOR IMAGES

[75] Inventor: Gar A. Bergstedt, Portland, Oreg.

[73] Assignee: Tektronix, Inc., Beaverton, Oreg.

[21] Appl. No.: 483,354

[22] Filed: Apr. 8, 1983

[51] Int. Cl.⁴ G09G 1/16

[52] U.S. Cl. 340/703; 340/709;

340/724

[58] Field of Search 340/701, 703, 709, 712,
340/711, 724

[56] References Cited

U.S. PATENT DOCUMENTS

3,846,826 11/1974 Mueller 340/703 X
4,180,805 12/1979 Burson 340/724 X
4,200,867 4/1980 Hill 340/724 X
4,232,311 11/1980 Agneta 340/703
4,367,464 1/1983 Kurahashi et al. 340/703 X
4,379,292 4/1983 Minato et al. 340/703 X
4,386,345 5/1983 Narveson et al. 340/703

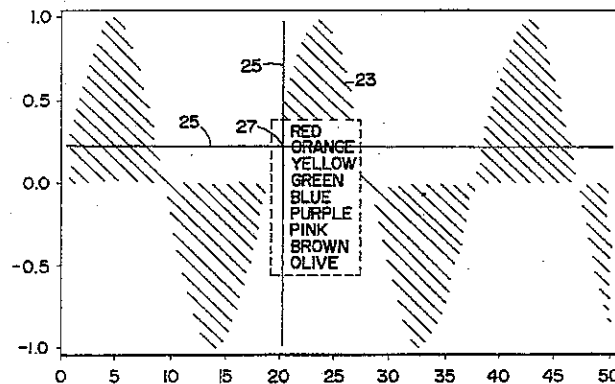
4,437,092 3/1984 Dean et al. 340/703
4,467,322 8/1984 Bell et al. 340/701

Primary Examiner—Marshall M. Curtis
Attorney, Agent, or Firm—Robert S. Hulse

[57] ABSTRACT

An apparatus and method are disclosed which permit selection of colors for display, and easy modification of displayed colors. The apparatus includes a microprocessor, a ROM with a stored program, a RAM with modifiable color data, a CRT display for displaying color images, and a keyboard with a cursor-positioning key for identifying a pixel of a displayed color image, and a color menu and H (hue), L (lightness), S (saturation) color modification keys. In response to actuation of one or more of the color modification keys, the microprocessor, under control of the stored program, modifies the stored color data, and uses the modified color data to control the color guns (red, green, blue) of the CRT to modify the color of the identified pixel.

7 Claims, 10 Drawing Figures



SET COLOR		F1	F2	F3	F4	F5
GRAPHICS	EXIT	HUE	LIGHTNESS	SATURATION	MAP	COLOR MENU
INDEX 3		240°	50%	100%	RESTORE COLOR	

U.S. Patent Sep. 15, 1987

Sheet 1 of 10

4,694,286

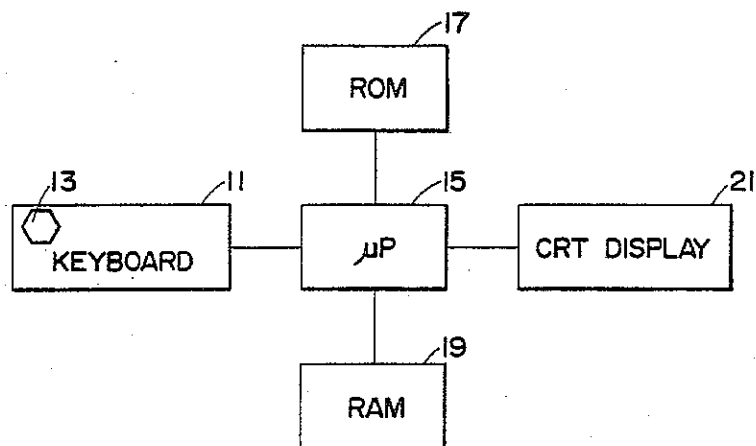
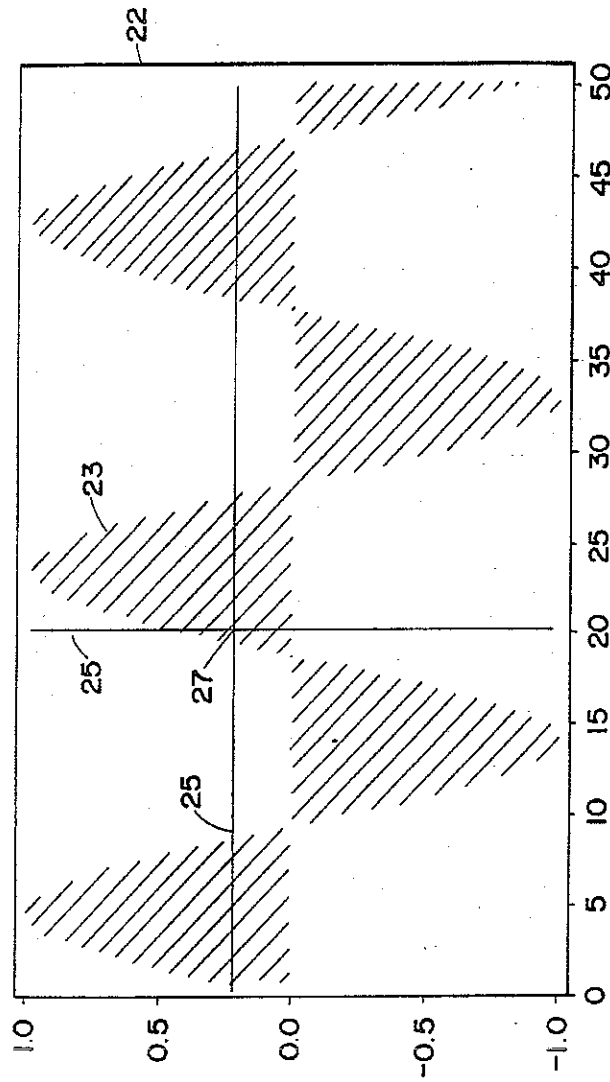


Fig. 1.

U.S. Patent Sep. 15, 1987

Sheet 2 of 10

4,694,286

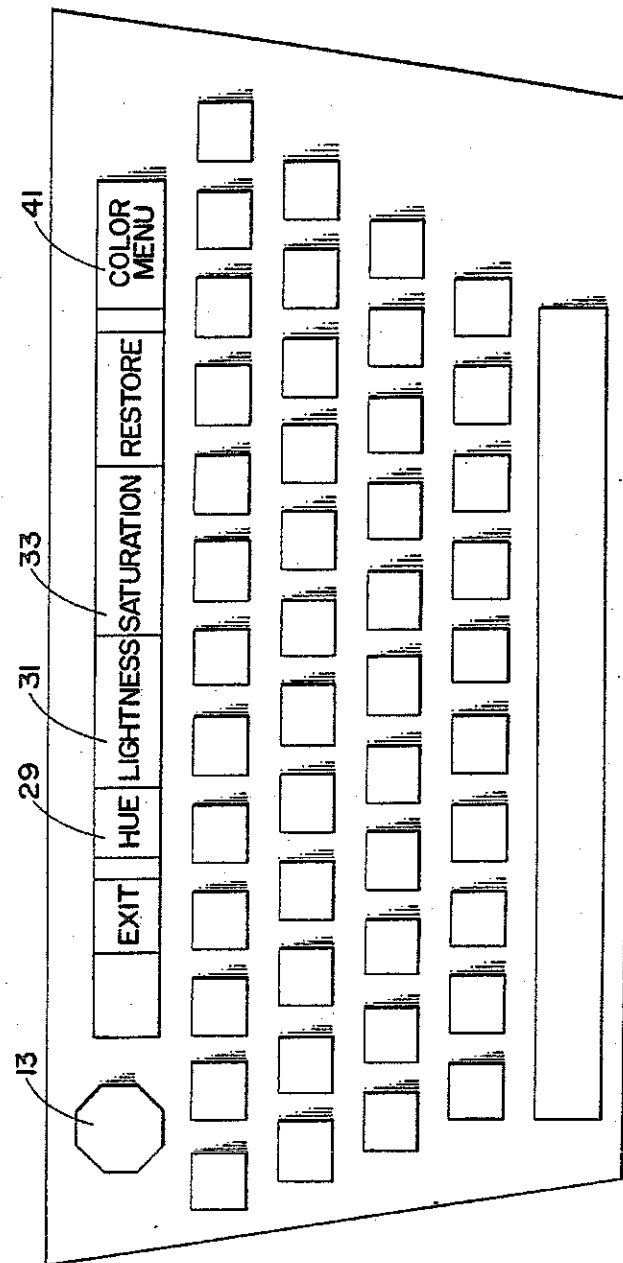
Fig 2

U.S. Patent Sep. 15, 1987

Sheet 3 of 10

4,694,286

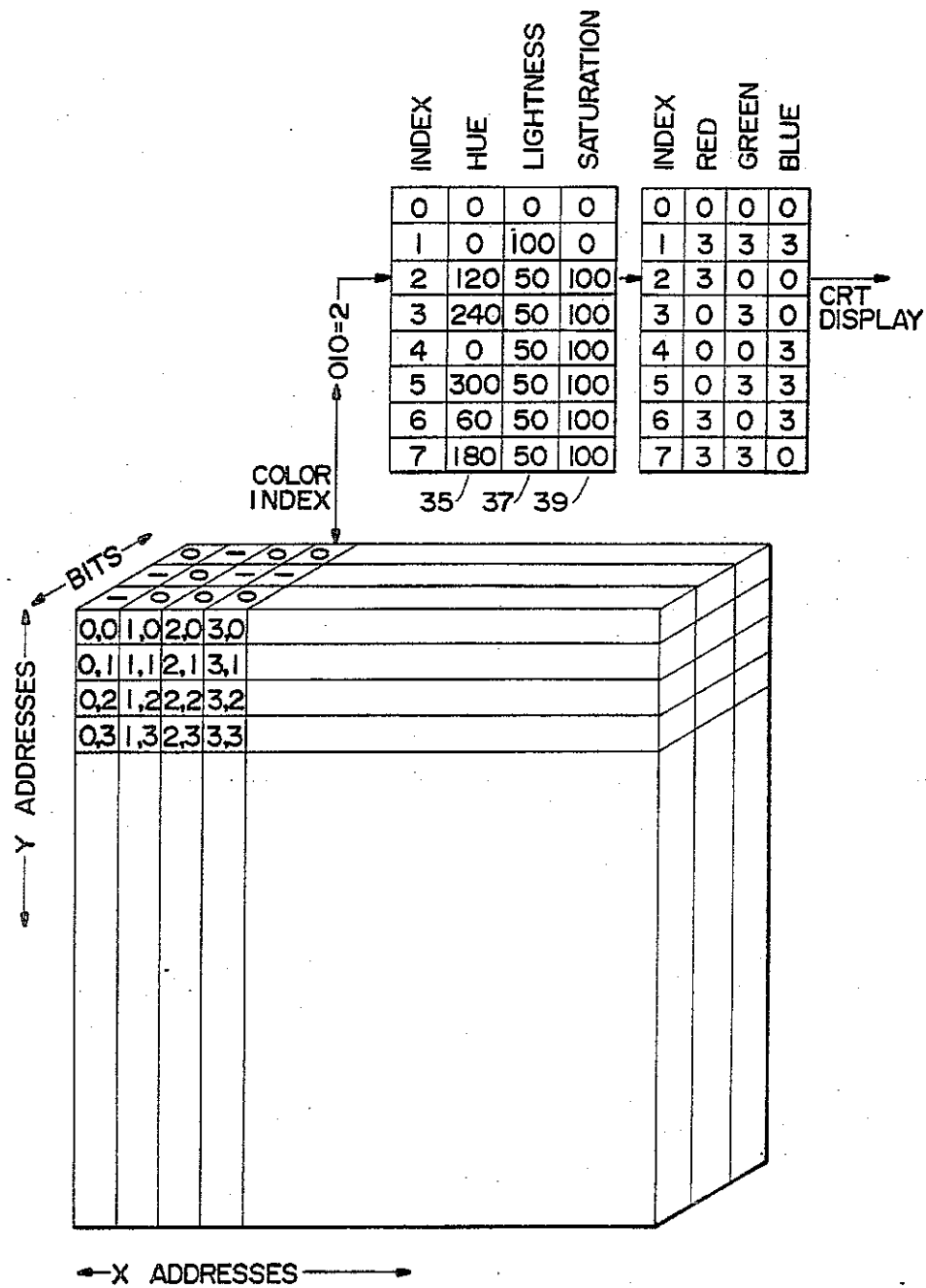
FIG 3.



U.S. Patent Sep. 15, 1987

Sheet 4 of 10

4,694,286

Fig. 4.

U.S. Patent Sep. 15, 1987

Sheet 5 of 10

4,694,286

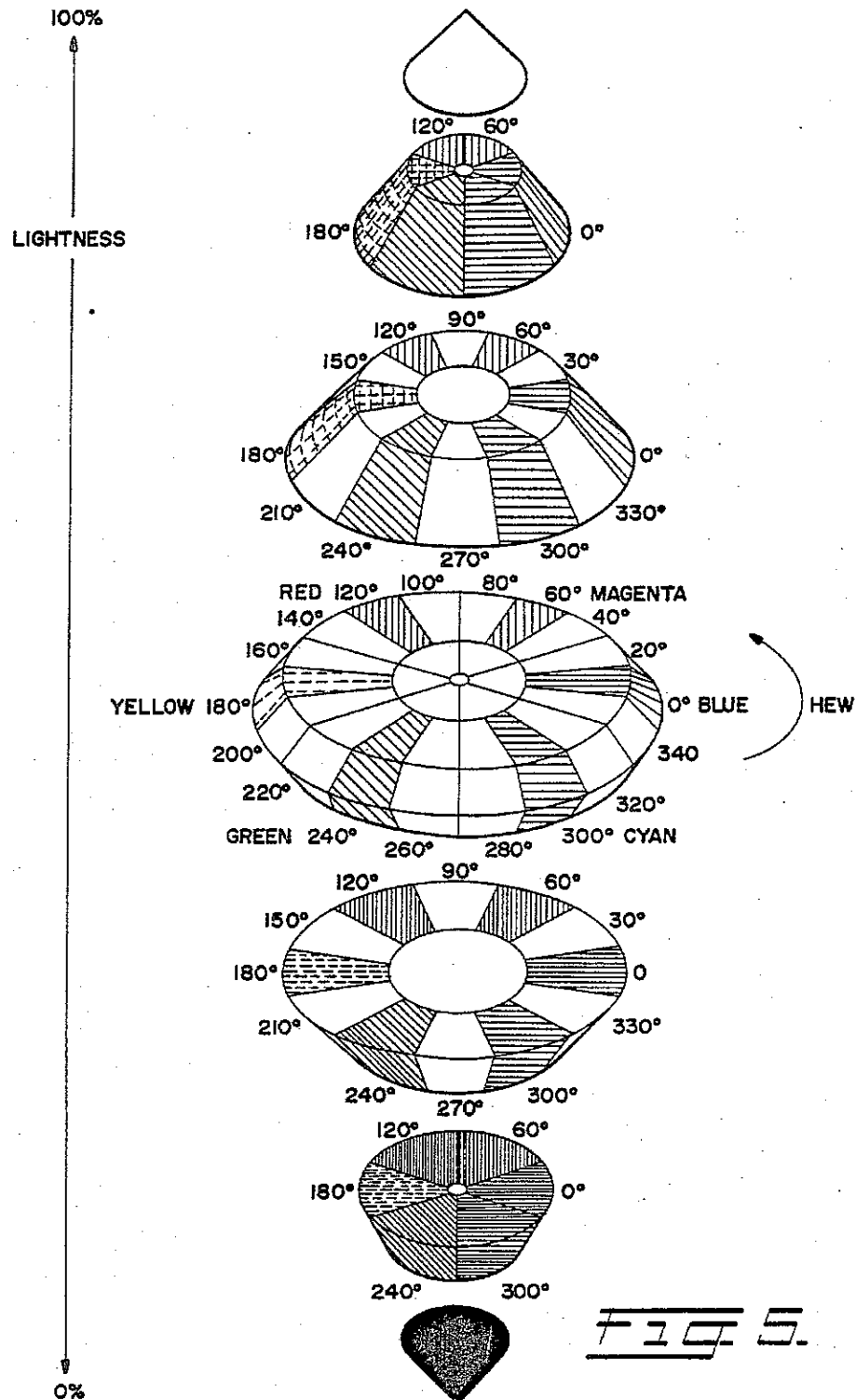
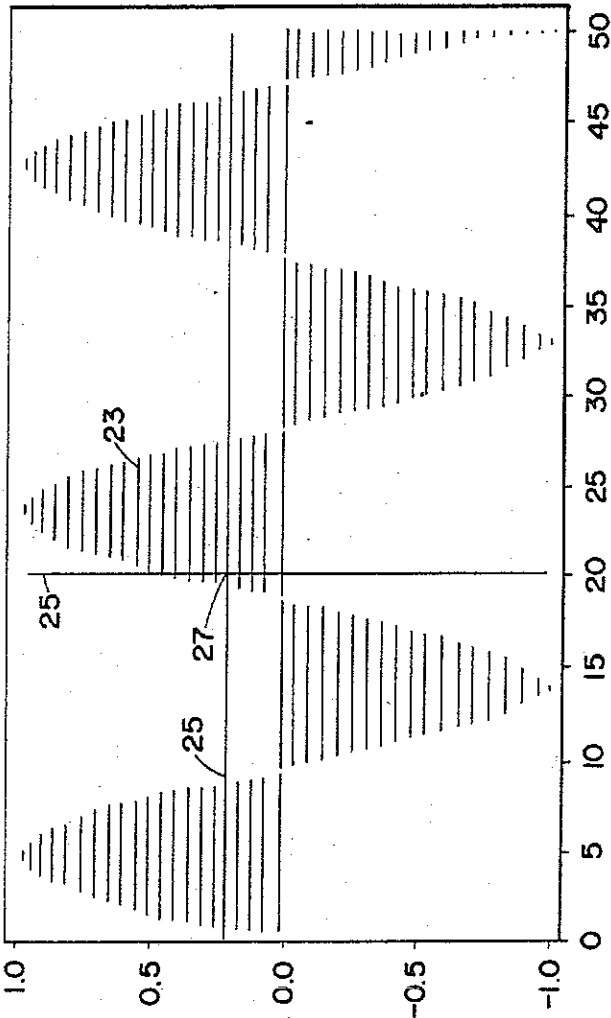
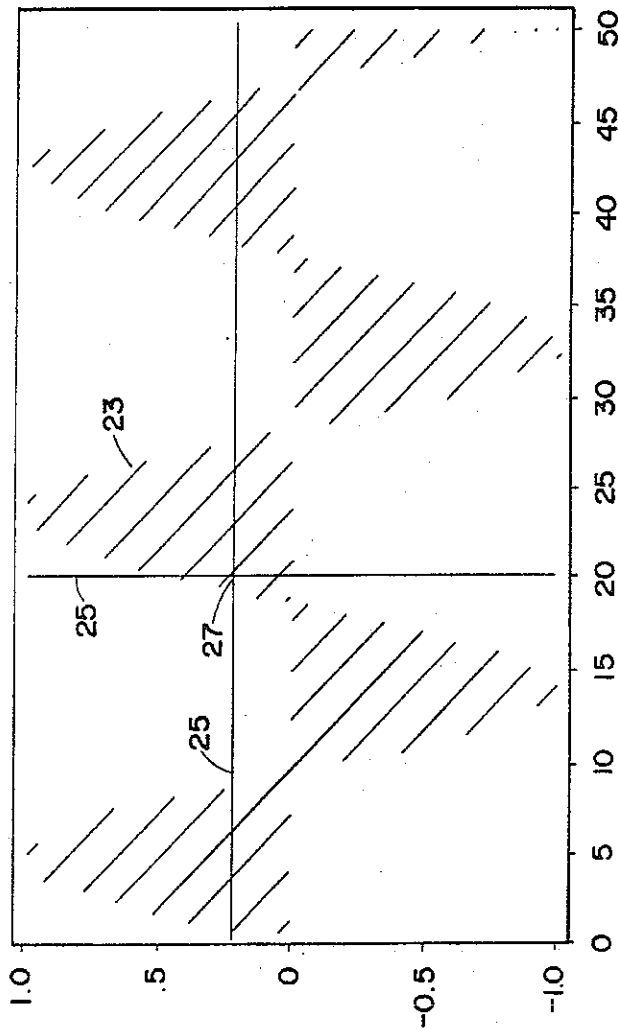


FIG. 6



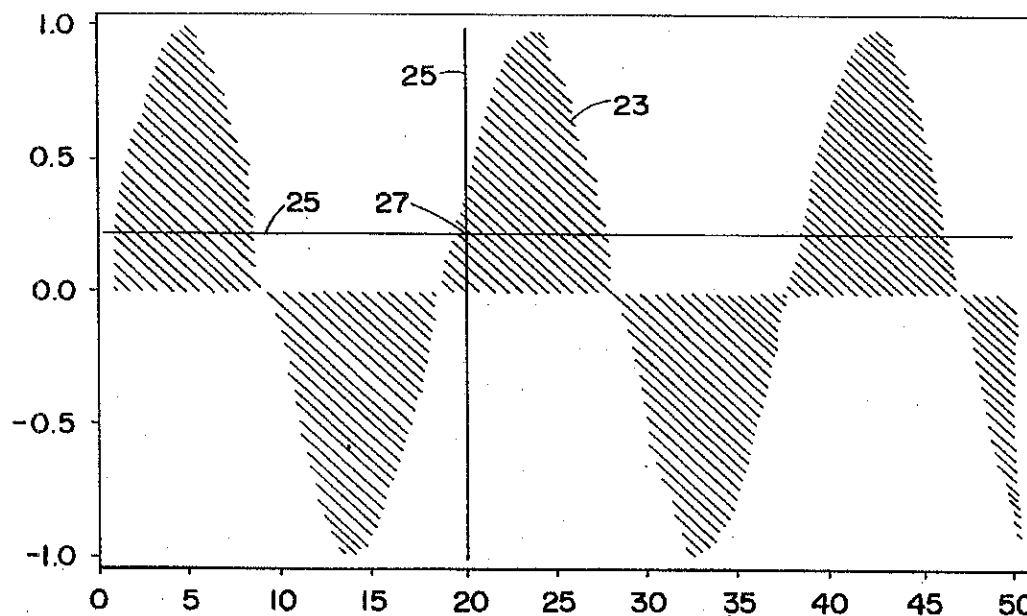
SET COLOR		F1	F2	F3	F4	F5
GRAPHICS	EXIT	HUE 300°	LIGHTNESS 50%	SATURATION 100%	MAP RESTORE COLOR	COLOR MENU
INDEX 3						

Fig. 7



SET COLOR		F1	F2	F3	F4	F5
GRAPHICS	<input checked="" type="checkbox"/>	HUE	LIGHTNESS	SATURATION	MAP	COLOR MENU
INDEX 3	EXIT	240°	80%	100%	RESTORE	COLOR

Fig. 8




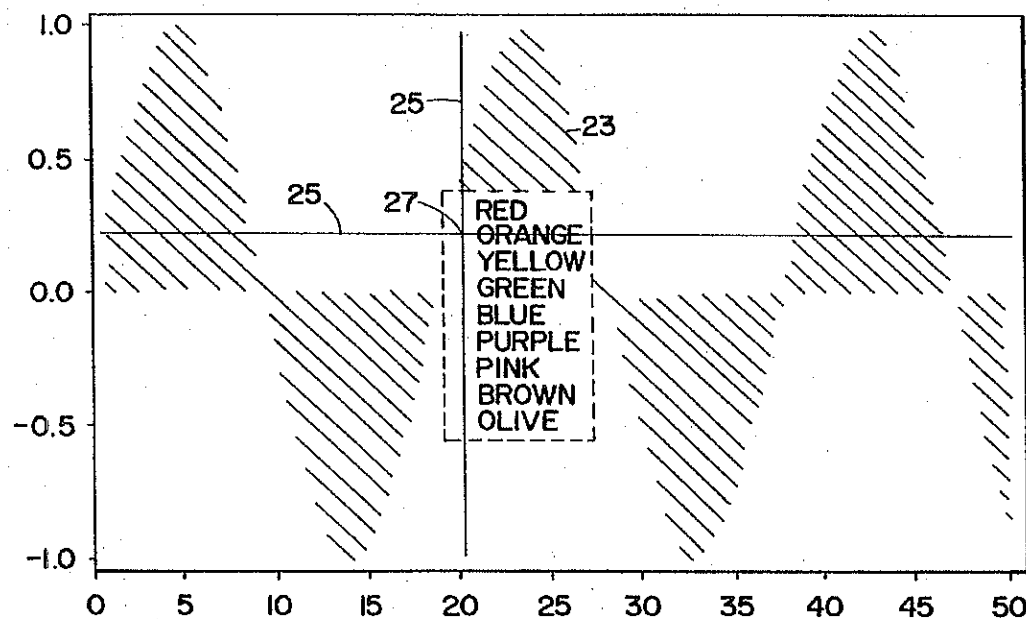
GRAPHICS  INDEX 3	SET COLOR	F1	F2	F3	F4	F5
	EXIT	HUE 240°	LIGHTNESS 50%	SATURATION 25%	MAP RESTORE COLOR	COLOR MENU

Fig 9

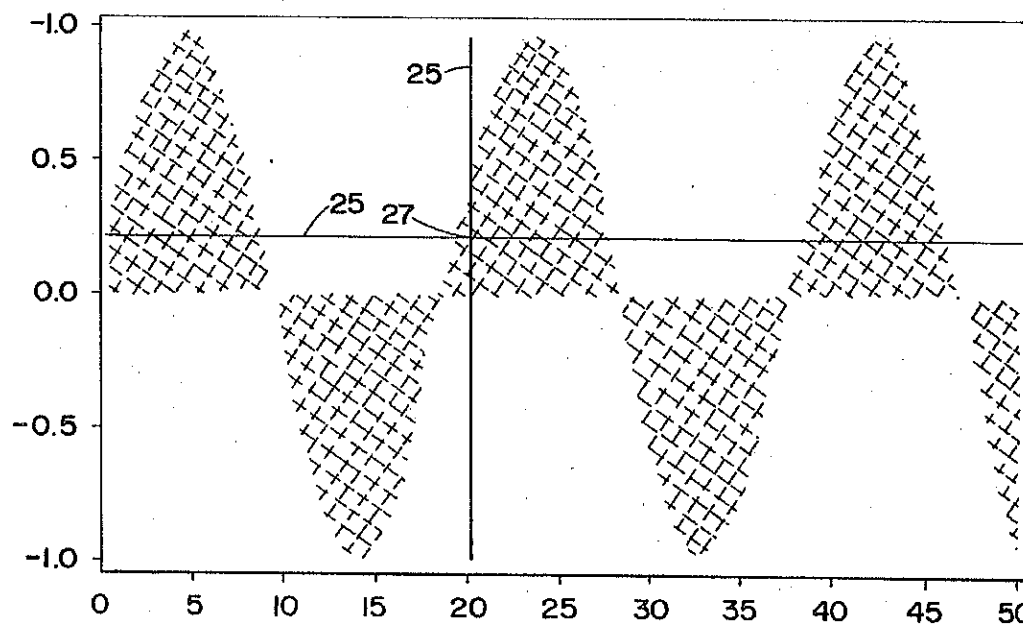


SET COLOR		F1	F2	F3	F4	F5
GRAPHICS	EXIT	HUE	LIGHTNESS	SATURATION	MAP	COLOR MENU
INDEX 3		240°	50%	100%	RESTORE COLOR	

U.S. Patent Sep. 15, 1987

Sheet 10 of 10 4,694,286

FIG. 10



SET COLOR		F1	F2	F3	F4	F5
GRAPHICS INDEX 3	EXIT	HUE 160°	LIGHTNESS 50%	SATURATION 100%	MAP RESTORE COLOR	COLOR MENU

1

APPARATUS AND METHOD FOR MODIFYING DISPLAYED COLOR IMAGES

BACKGROUND OF THE INVENTION

This invention relates generally to color display systems, and more particularly to color-selective graphic display terminals.

Many currently available display terminals require users to have a thorough understanding of color theory and how colors combine, and to enter substantial amounts of data before a desired color may be displayed. For example, one such terminal is the Megatek 1650 graphic terminal manufactured by Megatek Corporation of San Diego, Cal., U.S.A. To specify a color index and selected color coordinates (i.e., selected amounts of red, green, and blue base colors from which a resultant color or index is to be formed), such terminals often require the entry of four data values to specify the color index, followed by the entry of three more data values to specify the color coordinates; and even when the user satisfies these requirements, the specified color is often not displayed when its index is different from that of a previous image. To avoid this problem, the user often has to enter a display command; and, to do this, he is required to specify a color polygon, then build the polygon, coordinate by coordinate, superimposing the selected (desired resultant) color on the polygon—a multistep, often laborious process.

What is needed and would be useful, therefore, is a color selection and display apparatus which simplifies the process of color selection/specification.

SUMMARY OF THE INVENTION

Accordingly, an apparatus is provided which permits easy selection of colors for display, and easy modification of displayed colors. In accordance with the illustrated preferred embodiment of the present invention, the apparatus comprises a keyboard, a microprocessor, a ROM (read-only memory) with a stored program, a RAM (random-access memory) with predetermined, changeable color data, and a CRT (cathode ray tube) display for displaying color images. The keyboard includes a color menu key for initiating the display of a color menu, a disk key for selecting a color name from the menu or for specifying an image segment whose color is to be modified, and H (hue), L (lightness), S (saturation) keys for entering color modification data. In response to actuation of one or more of said keys, the microprocessor, under control of the stored program, modifies the color data and uses the modified color data to control the color (red, green, blue) guns of the CRT to produce a desired resultant color display.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of the apparatus of the present invention;

FIG. 2 is a graphic representation of an initial color image displayed by the apparatus of FIG. 1;

FIG. 3 is a top view of a keyboard of the apparatus of FIG. 1;

FIG. 4 is a graphic representation of a bit map arrangement of modifiable color data initialized by the apparatus of FIG. 1;

FIG. 5 is an exploded view of a color model having hue, lightness, and saturation coordinates corresponding to selected ones of the modifiable color data;

4,694,286

2

FIG. 6 is a graphic representation of the color image of FIG. 2 with the hue coordinate modified;

FIG. 7 is a graphic representation of the color image of FIG. 2 with the lightness coordinate modified;

FIG. 8 is a graphic representation of the color image of FIG. 2 with the saturation coordinate modified;

FIG. 9 is a graphic representation of the color image of FIG. 2 including a color menu; and

FIG. 10 is a graphic representation of the color image of FIG. 2 modified in accordance with selection of the orange color from the color menu of FIG. 9.

DESCRIPTION OF THE PREFERRED EMBODIMENT

Referring to FIG. 1, there is shown a system (apparatus) of the present invention comprising an alphanumeric keyboard 11 with cursor-positioning key or disk 13, microprocessor 15, read-only memory 17 for storing system control programs, random access memory 19 for storing variables such as color data, and CRT raster scan display 21 for displaying color in accordance with selected ones of the stored color data.

FIG. 2 shows a graphic image, in color (e.g., medium green), displayed on screen 22 of CRT display 21. To change the color of a segment (e.g., segment 23) of the image, the user of the system first positions cursor 25 at a pixel 27 (picture element) or portion of the segment to be changed, then actuates (depresses and releases) one or more of the hue, lightness, and saturation keys 29, 31, 33, respectively of keyboard 11 shown in FIG. 3. Cursor positioning is accomplished by actuation of disk key 13.

A color index is associated with each pixel. If the screen 22 is represented by an X-Y matrix of 480×360 pixels, for example, there will be correspondingly 172,800 indexes, one index associated with each pixel. An index is a series of bits which is used to address (point to) color coordinate values stored in RAM 19. For example, a series of three bits may be used to denote indexes 0-7, each index representing the starting address of a group of three color coordinate words 35, 37, 39, as shown in FIG. 4. The contents of the first, second, and third words 35, 37, 39 of each group represents hue data, lightness data, and saturation data, respectively, for the associated index. The hue, lightness, and saturation data correspond to spatial (angular, vertical, radial) coordinates of an HLS (hue, lightness, saturation) color cone as shown in FIG. 5. Upon successive actuations of hue key 29, the color of the image segment at cursor position 27, and all other segments having the same color index, is changed discretely (e.g., each actuation corresponding to a 20° change) in a clockwise manner around the cone (i.e. 0 to 360 degrees) as shown in FIG. 5. For example, when hue key 29 is actuated three times from the hue value of 240° shown in FIG. 2, the hue value is changed from 240° to 300° , causing the color of all image segments having the same index as that of the pixel at the middle green color of FIG. 2 to change to the cyan color of FIG. 6. Likewise, upon successive actuations of lightness key 31, the color of the image segment at cursor position 27, and all other segments having the same color index, is changed discretely (e.g., each actuation corresponding to a 10% change) from 0% to 100% vertically up the cone as shown in FIG. 5. From 100%, further actuation of key 31 causes color lightness to start varying again from 0% (black) toward 100% (white). For example, when lightness key 31 is actuated three times from the lightness value of 50%

4,694,286

3

shown in FIG. 2, the lightness value is changed from 50% to 80%, causing the color of all image segments having the same index as that of the pixel at cursor position 27 to change from the medium light green color of FIG. 2 to the lighter green color of FIG. 7. Similarly, upon successive actuations of saturation key 33, the color of the image segment at cursor position 27, and all other segments having the same color index, is changed discretely (e.g., each actuation corresponding to a 25% change) from 0% at the axis of the cone to 100% toward the periphery of the cone, as shown in FIG. 5. From 100%, further actuation of key 33 causes color saturation to start varying again from 0% (no saturation) toward 100% (maximum saturation). For example, when saturation key 33 is actuated twice from the saturation value of 100% shown in FIG. 2, the saturation value is changed from 100% to 0% to 25%, causing the color of all image segments having the same index as that of the pixel at cursor position 27 to change from the maximum saturated green color of FIG. 2 to less saturated green color of FIG. 8.

Whenever the hue, lightness, and saturation values are changed for a given index (e.g., for the index three associated with the pixel at cursor position 27 shown in FIGS. 2 and 6 to 8) because of actuation of the hue, lightness and saturation keys 29, 31 and 33, the changed values are stored as new hue, lightness and saturation coordinate values in words 35, 37, and 39 for that index. As shown in FIGS. 2 and 6-8, the coordinate values and associated index indicative of a particular color are displayed for the user's convenience if the user wishes to specify the same color at a later time. To do so, the user merely enters, via keyboard 11, the coordinates and index for that color.

The foregoing represents one method of changing color of image segments. The following represents another method, utilizing a color menu. Referring to FIG. 2, there is shown a displayed image with cursor positioned at a pixel 27, the color of which pixel 27 (and associated segment 23) the user desires to change. FIG. 3 shows keyboard 11 having a color menu key 41. Upon depression of key 41, a color menu 43 is displayed at a predetermined distance proximate to the cursor. To select a color from the menu and to have the color of the pixel at which the cursor is positioned change to the selected color, the user simply moves the cursor, by repeated actuations of disk key 13, and positions it on the menu color of his choice as shown, for example, in FIG. 9. Thereafter, upon release of color menu key 41, the color of the pixel (and of all image segments having the same color index as that of the pixel) is changed to the color selected from the menu, and display of the menu is terminated, as shown in FIG. 10. Each color of the menu is represented by three HLS color coordinate values. These coordinate values, which are predetermined, are stored in ROM 17.

Thus, the color menu key and the HLS keys may be viewed as color modification devices. After the HLS coordinate values are changed in response to actuation of one of the HLS keys, or actuation of the color menu key, the changed values are converted by known meth-

4

ods to RGB (red, green, blue) values which are used by the CRT display to regulate the output of its RGB guns.

I claim:

1. An apparatus for modifying color characteristics of a displayed color image formed by a plurality of pixels, the apparatus comprising:

means for storing pixel color data, including index values and sets of hue (H), lightness (L) and saturation (S) color values, each index value associating one color, represented by one set of H, L, S color values, with at least one pixel;

means for selecting a color of the image for modification by positioning a cursor at a specified pixel having that selected color; and

means for modifying at least one of the set of H, L, S color values associated with the specified pixel in response to input of new color data, said modifying means including a means for inputting new color data for at least one of said set and a processor means responsive to the new color data for changing said at least one of the set of stored H, L, S color values to the new color data, enabling color modification, according to the modified set, of all pixels of the image having the same index value as the specified pixel.

2. An apparatus as recited in claim 1 wherein the input means includes a means for entering index values.

3. An apparatus as recited in claim 1 wherein the input means is a keyboard with a plurality of keys including H, L, S keys for entering H, L, S color values for specified pixels, each value being changeable by a predetermined discrete amount upon each actuation of the respective H, L, S key.

4. An apparatus as recited in claim 1 wherein the input means includes:

a menu key, in response to actuation of which a menu of color items is displayed; and wherein the cursor is positionable at an item of the menu for selecting a color from the menu for display.

5. A method for modifying selected colors of a displayed color image formed by a plurality of pixels, the method comprising the steps of:

selecting a color of the image to be changed, by positioning a cursor at a selected pixel having that color, each pixel having an index associated with one set of a plurality of sets of stored hue (H), lightness (L), and saturation (S) color values;

inputting at least one new color value; and

changing at least one of the stored H, L, S color values of the set associated with the particular index of the selected pixel, enabling the color of all pixels having said particular index to be changed to the color corresponding to the set of changed color values.

6. A method as recited in claim 5 wherein said inputting step comprises the step of actuating at least one input key for entering hue, lightness, and saturation color values.

7. A method as recited in claim 5 wherein said inputting step includes the steps of displaying a menu of colors and selecting a color from the menu.

* * * * *

United States Patent [19]

Huntzinger

[11] Patent Number: 4,761,642

[45] Date of Patent: Aug. 2, 1988

- [54] SYSTEM FOR PROVIDING DATA COMMUNICATION BETWEEN A COMPUTER TERMINAL AND A PLURALITY OF CONCURRENT PROCESSES RUNNING ON A MULTIPLE PROCESS COMPUTER

[75] Inventor: Gregory G. Huntzinger, West Linn,
Oreg.

[73] Assignee: Tektronix, Inc., Beaverton, Oreg.

[21] Appl. No.: 784,413

[22] Filed: Oct. 4, 1985

[51] Int. Cl.⁴ G09G 1/16

[52] U.S. Cl. 340/721; 340/723;
340/747; 364/521

[58] Field of Search 340/707, 721, 747, 724,
340/723, 750; 364/521

[56] References Cited

U.S. PATENT DOCUMENTS

3,898,643	8/1975	Ettlinger	340/707
4,258,361	3/1981	Hydes et al.	340/721
4,498,081	2/1985	Fukushima et al.	340/721
4,559,333	12/1985	Bass et al.	340/747
4,598,384	7/1986	Shaw et al.	340/721

4,618,859	10/1986	Ikeda	340/747
4,642,790	2/1987	Minshull et al.	340/747

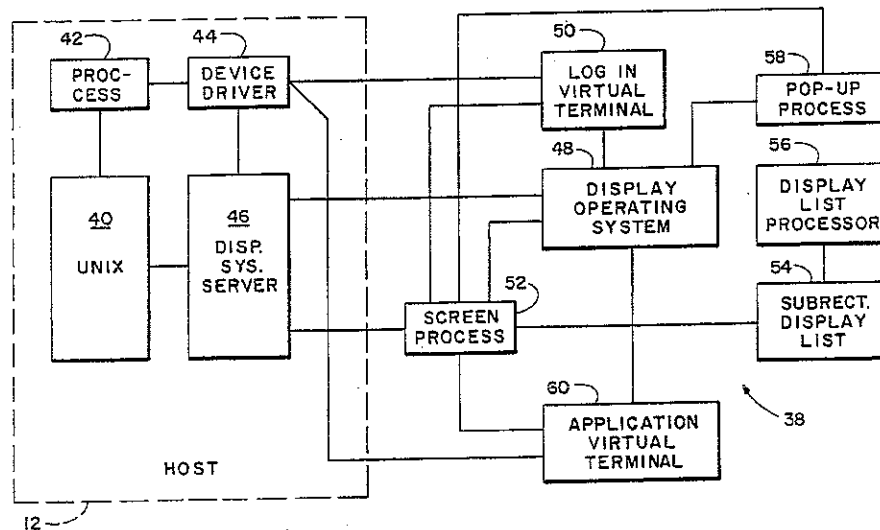
Primary Examiner—Marshall M. Curtis

Attorney, Agent, or Firm—Daniel J. Bedell; Robert S. Hulse

[57] ABSTRACT

An apparatus for displaying the display outputs of a plurality of simultaneously active computer processes in corresponding windows on a single screen includes a microcomputer, a display screen and display system software. The software represents a plurality of virtual terminals, one corresponding to each active process, for emulating the operation of real terminals communicating with the processes. Each virtual terminal maintains a display list comprising a set of instructions defining a display on a screen according to the output from the corresponding process. The software also includes a screen process for maintaining a subrectangle list comprising a set of instructions for allocating window portions of the screen to the displays defined by the separate display lists. A display list processor is provided for creating the windows on the screen according to the display and subrectangle lists.

1 Claim, 5 Drawing Sheets

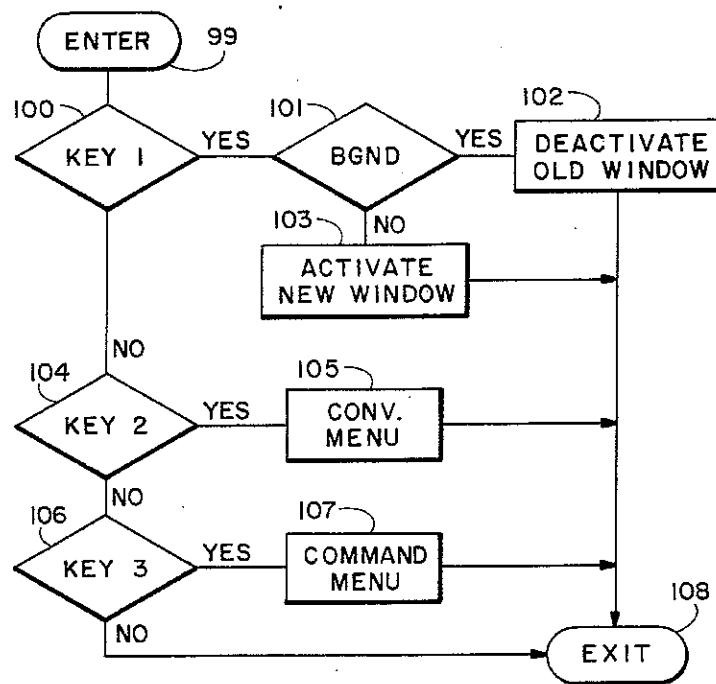
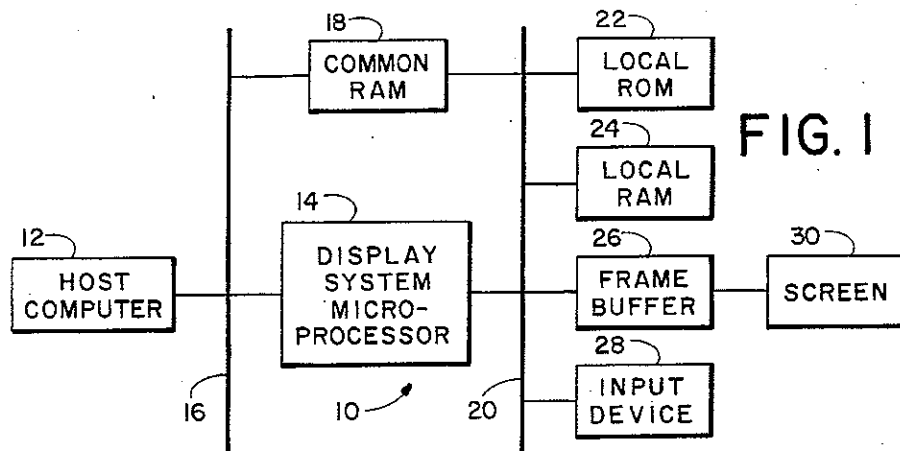


U.S. Patent

Aug. 2, 1988

Sheet 1 of 5

4,761,642



U.S. Patent Aug. 2, 1988

Sheet 2 of 5

4,761,642

CREATE
DESTROY
REFRAME
MOVE
COLLAPSE
EXPAND
BURY
UNCOVER

FIG. 2

REDRAW
BLOCK
LOG IN
HARD COPY
SOFT COPY
SET ATTRIB.

FIG. 3

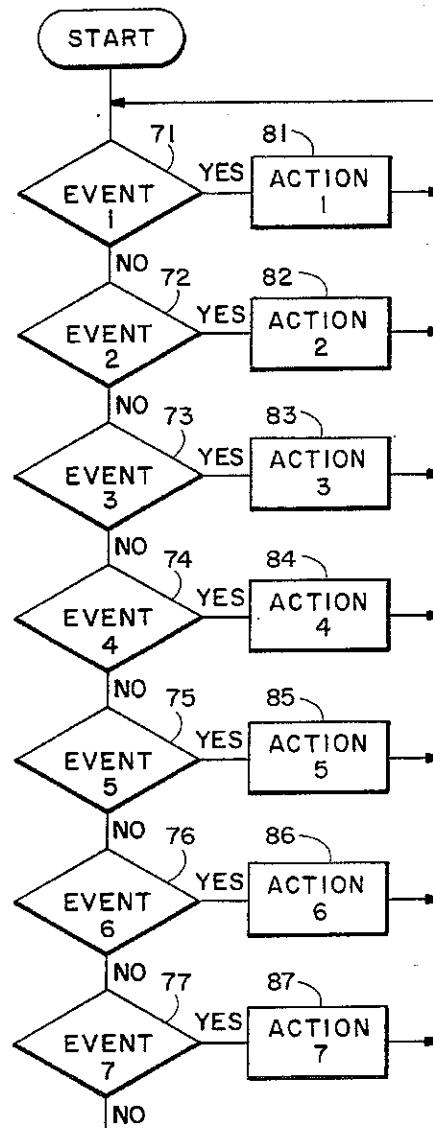


FIG. 5

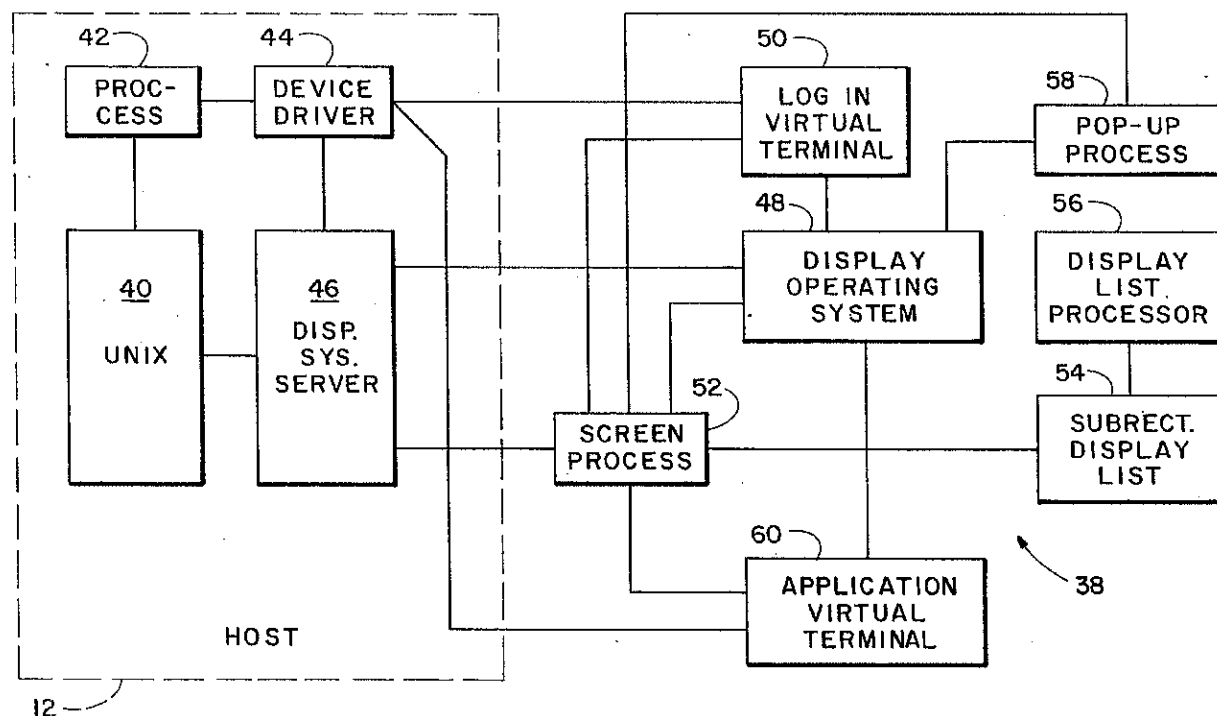


FIG. 4

U.S. Patent Aug. 2, 1988

Sheet 4 of 5

4,761,642

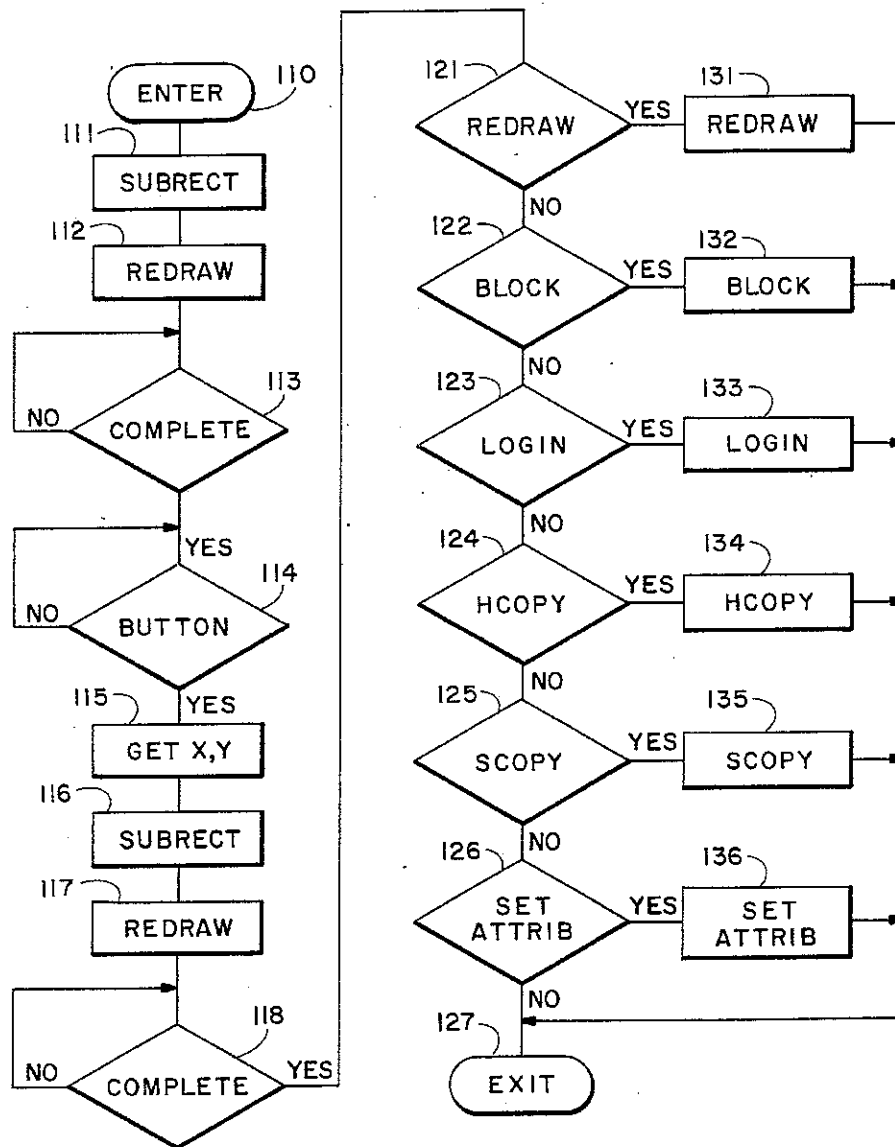


FIG. 7

U.S. Patent

Aug. 2, 1988

Sheet 5 of 5

4,761,642

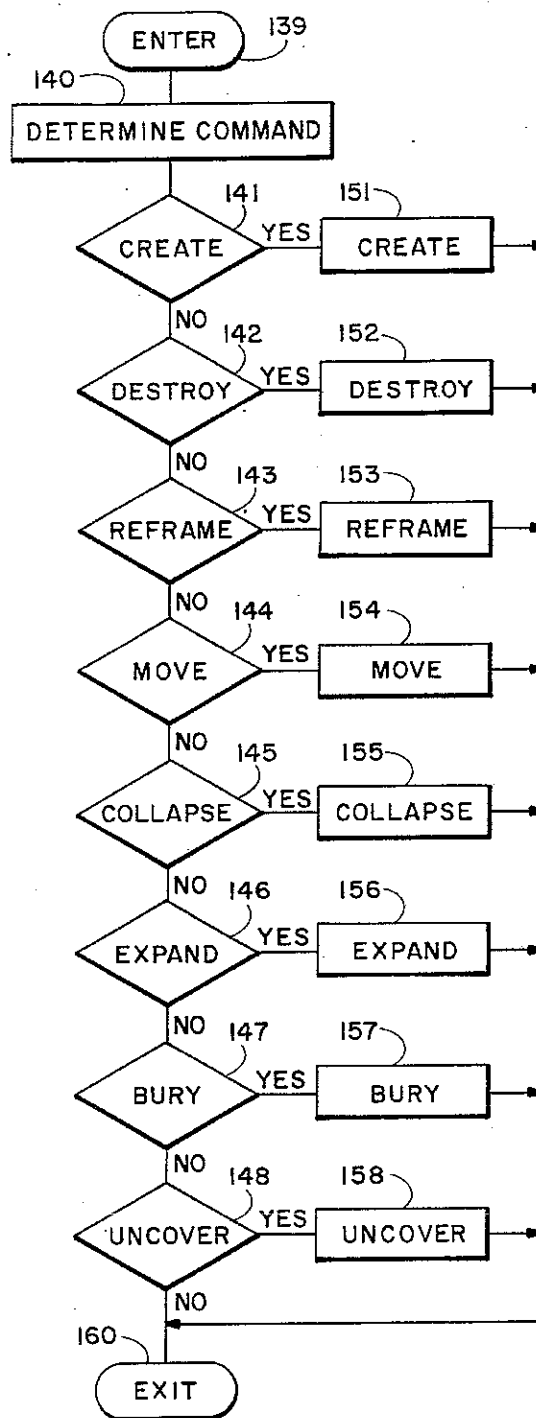


FIG. 8

4,761,642

1

SYSTEM FOR PROVIDING DATA COMMUNICATION BETWEEN A COMPUTER TERMINAL AND A PLURALITY OF CONCURRENT PROCESSES RUNNING ON A MULTIPLE PROCESS COMPUTER

BACKGROUND OF THE INVENTION

The present invention relates in general to multiple process computer systems and more particularly to a method and apparatus for simultaneously displaying multiple process outputs on a single screen.

Typically operators access computers through terminals including an input device such as a keyboard and an output display device such as a cathode ray tube (CRT) screen. Although many types of terminals are available, terminals of the prior art provide for a single output stream directed from the computer to the operator and a single input stream from the operator to the computer. When the terminal accesses a multiprocess computer, only one computer process attaches its input and output to these streams and does not relinquish them to another computer process until it is suspended. At that time another computer process may attach its input and output to these streams.

Most operators would find it difficult to manage more than one input device simultaneously so the limitation that a terminal can handle only a single input stream is of little practical significance. However, operators can monitor more than one output stream and the limitation that the terminal can handle only one output stream is more serious. For instance, an operator in a power plant may wish to view the outputs of several programs running on a multi-process computer which monitors plant operation. If only one terminal is used, the operator could only view the program outputs in succession since only one program can output to the terminal at a time. The traditional alternative is to provide a separate terminal for each process output stream. This permits the display outputs of all of the processes to be updated continuously and simultaneously.

The use of multiple terminals is an expensive solution to the problem, in terms of space and money, and it is often inconvenient for the operator to simultaneously monitor several screens. A partial remedy to this problem found in the prior art involves the use of separate windows on a CRT screen to display the outputs of separate computer processes. But there are limitations to this solution as well. First, there is still only one output stream and therefore only one process can update its associated display window at a time. The other processes must wait until the process controlling the stream is finished with its current window update cycle and relinquishes the output stream. Therefore, while the outputs of several processes may be displayed on a single screen, only one such output can be active and a process which must send data to the terminal before it can continue is suspended until it acquires control of the output stream. Secondly, to provide for larger views, when given a limited screen size, the windows are typically overlapped with the active window on top. When a process takes over the stream, its window is placed on top while portions of other windows which are covered are lost. This problem has been remedied, in the case of a terminal using a frame buffer memory storing display control data, by moving the data representing obscured window portions from the frame buffer memory to a secondary storage memory so that the obscured por-

2

tions of the windows can be restored when these windows are once again brought to the forefront. However, such movement of display data takes time, retarding the speed of screen update.

5 A third problem associated with existing window systems relates to the need to modify each application process so that it writes only to the window and not the entire screen. This makes it difficult to adapt preexisting software for use in conjunction with a windowed display system.

10 What is needed is a system whereby output streams from multiple, active, application processes can be directed from a computer to a single display screen for concurrent, active display without requiring modification to the application programs.

SUMMARY OF THE INVENTION

According to one aspect of the invention, a display system is provided for transmitting and receiving the input and output streams of each of a plurality of processes running on a multiple process, multiple user host computer. Each input stream carries input data from an operator to the process and each output stream carries screen update data from the process to the display system. The display system includes a microprocessor for running a multiple process display operating system. For each output stream, the display operating system creates an associated "virtual terminal" process for sending and receiving the input and output streams of the associated host computer process. Each virtual terminal is adapted to simulate the operation of a selected real terminal with respect to the transmission and receipt of input and output streams, except that each virtual terminal merely prepares and stores a set of instructions (a display list) for creating a full screen display according to the data from the associated process, but does not independently control a separate screen. The display lists maintained by each virtual terminal are sent to a common display list processor which creates a set of windows on a single screen, each window containing a display according to a separate one of the display lists. Thus a single terminal appears to the host computer to be a set of multiple terminals. Each host computer process has sole use of an input/output stream and it is not necessary to suspend one host computer process and activate another in order to provide shared access to a terminal.

According to another aspect of the invention, the display window associated with each host computer process may be created, deleted, moved, buried beneath another window, uncovered, collapsed or expanded according to commands from an operator. A screen process created by the display operating system monitors instructions from the operator regarding these window control operations and creates a "subrectangle list" indicating which windows are to be displayed along with the size of each window, the positioning of the window on the screen and the portions of which windows are to be obscured by overlapping windows. As new display lists are produced by each virtual terminal, they are provided to the display list processor which converts the display list data into display control data transmitted to a frame buffer that updates the display. The virtual terminal associated with each window updates, stores and transmits display lists to the display list processor each time it receives data from the host process, regardless of whether the window is currently

4,761,642

3

displayed or whether some portion of the window is obscured. The display list processor uses the subrectangle list to determine what portions of each window are to be displayed, and uses the display lists to determine the nature of the display. Thus every displayed window is output active in the sense that it may be changed by the associated host computer process at any time regardless of the input/output operation of any other host computer process.

According to a further aspect of the invention, each time the operator makes a change in the way one or more windows are displayed, the screen process changes the subrectangle list and transmits a redraw command to each virtual terminal associated with a changed window. Each such virtual terminal retrieves its display lists from memory and transmits them to the display list processor which then modifies the screen according to the display and subrectangle lists. Thus it is not necessary for data to be moved from the frame buffer memory to a secondary memory in order to save screen control data corresponding to portions of a window obscured by another window or temporarily removed from view. Therefore, windows not currently displayed or only partially displayed are still output active in the sense that the display lists controlling the windows are independently updated whenever a host computer process transmits output data to its associated virtual terminal. Since the virtual terminals store these lists in memory, the lists are readily available to the display list processor whenever the operator chooses to display the associated window. Also, since each process in the host computer has exclusive access to a corresponding virtual terminal maintaining a display list including instructions for writing to an entire screen, it is not necessary to modify the process application so that the process writes only to a window portion of a screen.

According to still another aspect of the invention, while every window is simultaneously output active, only a single displayed window is input active in that data transmitted to the terminal from the operator using a keyboard or other input device is forwarded only to a virtual terminal associated with a single, selected window for further transmission to the associated host computer process. The display system is adapted to permit an operator to select the window to be input active by placing a cursor over the window and operating a pushbutton. This feature permits the operator to provide input data to any one of several concurrent host computer processes from a single terminal and to rapidly redirect input data to a different host computer process without suspending one process and activating another.

It is accordingly an object of the present invention to provide a new and improved method and apparatus for providing concurrently active output displays from multiple processes on a single screen.

It is another object of the invention to provide a new and improved method and apparatus for providing operator input to any selected one of a plurality of processes from a single input device.

It is a further object of the invention to provide a new and improved method and apparatus for displaying outputs of multiple processes in screen windows when the processes are adapted for writing to entire screens.

The subject matter of the present invention is particularly pointed out and distinctly claimed in the concluding portion of this specification. However, both the organization and method of operation of the invention, together with further advantages and objects thereof,

4

may best be understood by reference to the following description taken in connection with accompanying drawings wherein like reference characters refer to like elements.

DRAWINGS

FIG. 1 is a hardware block diagram of a multiple process windowed display system of the present invention;

FIG. 2 is an illustration of a pop-up command menu which may be displayed by the terminal of FIG. 1;

FIG. 3 is an illustration of a pop-up convenience menu which may be displayed by the terminal of FIG. 1;

FIG. 4 is a software block diagram of the multiple process, windowed display system of the present invention;

FIG. 5 is a flow chart of a software state machine for controlling the virtual terminal and screen processes of FIG. 2;

FIG. 6 is a flow chart of software implementing a portion of the screen process of FIG. 2;

FIG. 7 is a more detailed flow chart of software implementing a block of the flow chart of FIG. 6; and

FIG. 8 is a more detailed flow chart of software implementing another block of the flow chart of FIG. 6.

DETAILED DESCRIPTION

Referring to FIG. 1, there is depicted in block diagram form a display system 10 adapted to provide input and output access to a multiple process, multiple user host computer 12. The display system 10 comprises a display system microprocessor 14 connected to the host computer 12 by a common bus 16. The host computer 12 and the display system microprocessor 14 communicate with one another by reading and writing data to a common random access memory (RAM) 18. The system 10 also includes a local bus 20 providing communication between the microprocessor 14 and a local read only memory (ROM) 22, a local RAM 24, a frame buffer 26 and an operator input device 28 including a keyboard and a pushbutton mouse. Frame buffer 26 controls a display screen 30.

Host computer 12 suitably operates under the UNIX operating system which permits the computer to simultaneously run multiple independent processes and to provide input and output interface between each process and an associated terminal in a known fashion. System 10 is adapted to emulate the operation of a plurality of terminals, each independently managing input and output streams from a corresponding process running in host computer 12. The output of each process is selectively displayed in a corresponding window on screen 30, while operator input to each process is provided through a common input device, keyboard and mouse 28. Each window displayed on screen 30 is output active in that it may be changed according to data provided by the host computer process irrespective of the current input/output activity of any other process. However, only a selected one of the display windows is input active such that operator input from the keyboard or mouse 28 of input device is forwarded to only a single, selected host computer process at a time. The operator selects the window to be input active by using the mouse to move a cursor over the window and then pressing a first button on the mouse. Thereafter, all data from the input device 28 is forwarded to the associated host computer process until another window is selected.

4,761,642

5

The operator may create or destroy processes by creating or destroying windows, may hide a window from view, and may change the relative positions of windows on the screen, the size of the windows, and the order in which windows overlap. To do so, the operator places the cursor at a selected location on screen 30 and then presses and holds a second button on the mouse to cause a pop-up command window 34, depicted in FIG. 2, to be displayed on the screen at the selected location. The command window contains several boxes, each representing a separate command. The operator then moves the cursor to the selected command box and releases the button, causing the command window to disappear and the selected command to be executed. A "create" command permits the operator to create a window. A corner shaped cursor appears on the screen and the operator moves it to the position on the screen where the upper left hand corner of the new window is to be located and presses a mouse button. The operator then moves the cross hair cursor to a position on the screen where the lower right hand corner of the window is to be located and again presses the mouse button. The display system 10 then transmits information to the host computer operating system that a new process is requested and the host computer operating system creates the new process. Display system 10 creates a blank window on screen 32 at the corner coordinates defined by the operator and also creates a software-based "virtual terminal" to provide a point of interface between the new process and the terminal. The virtual terminal emulates the operation of a real terminal which the new process is capable of driving. The particular terminal to be emulated is determined by the operator's responses to prompts displayed on the screen following selection of the create command. Subsequent data transmitted from the new process to the new virtual terminal is used to control the display within the new window. Thus system 10 maintains a separate virtual terminal to service the input/output requirements of each independent process in the host computer 12 and each virtual terminal controls the display within a corresponding window on screen 30.

The operator can destroy a window by popping up the command window and selecting a "destroy" command. This command causes the virtual terminal associated with the host computer process to transmit a process termination message to the host computer and also causes system 10 to subsequently terminate the virtual terminal associated with the window and remove the window from the screen.

The operator can also select a "reframe" command from the command window permitting him to redefine the size and position of an existing window in the same way he defined the size and position of a new window. The reframed window is then displayed on the screen while the existing window is collapsed. A "move" command permits the operator to "drag" an existing window from one screen location to another by selecting the window with the cursor and moving the cursor to a new location before releasing a cursor button. A "bury" command permits an operator to place a selected screen "behind" another window in the manner that any overlapping portion of the selected window is obscured by the other window. An "uncover" command has the opposite effect, permitting the operator to select a window to be placed in front of an overlapping portion of any other windows. An "activate" command permits the operator to select a window to be input activated.

6

This command has the same effect as directly selecting a window with the cursor and pressing an activate button on the mouse, as discussed hereinabove. A "collapse" command permits an operator to temporarily remove a selected window from display without destroying the associated process or virtual terminal. In such case the associated virtual terminal continues to receive, process and store display data from the host computer process but the window is not displayed. The display system 10 creates and displays a small icon representing the collapsed window along one edge of the screen. An "expand" command on the command menu permits the operator to restore to the screen a window which has been removed by the collapse command by selecting the appropriate icon.

Using another button on the mouse, the operator may call another pop-up window, the convenience window 36 illustrated in FIG. 3 which permits the operator to select additional commands. A "redraw" command causes the terminal to display all windows, including those previously collapsed. A "block" command permits the operator to prevent the terminal from updating the display of any window until a password is typed into the terminal using the keyboard. A "log in" command causes the terminal to display a "log in" pop-up window. Display system 10 creates the log in window, along with an associated virtual terminal, when system 10 is booted to provide the operator with access to the host computer operating system for logging into and out of the host generating system. The log in window can be collapsed like any other window but cannot be destroyed. "Hardcopy" and "softcopy" commands on the convenience menu permit the operator to send the current state of a selected window to a printer or to a disk file. A "set attribute" command permits the operator to set or change various display attributes of a selected window such as background and foreground colors, font style and the like by answering screen prompts with the keyboard.

Referring to FIG. 4, there is depicted a software block diagram of the multiple process windowed display system 38 of the present invention, along with a software block diagram of the host computer 12 served by the display system. The host computer 12 of FIG. 1 suitably operates under the UNIX operating system 40 adapted to simultaneously execute several applications programs by setting up separate processes 42 for each program. A device driver 44 manages input and output data streams between each process and an associated external terminal. A display system server 46 controls the routing and formatting of these input and output data streams between the device driver and the terminals.

The display system 38 includes a display operating system 48 to control the operation of microprocessor 14 of FIG. 1. When the system 10 is booted, the display operating system 48 is loaded into memory and implemented by microprocessor 14. The display operating system 48 is also a multiple process operating system and it initially creates a log in virtual terminal process 50 to communicate with the UNIX operating system 40, sending data to the display system server 46 indicating the nature of the terminal emulated by the virtual terminal and the software I/O socket at which it is located. The display system server then provides the appropriate data to device driver 44 to establish a communication path between the UNIX operating system 40 and the virtual terminal 50.

4,761,642

7

Also following display system boot, the display operating system 48 establishes a screen control process 52 which controls the display of windows on the screen by maintaining a "subrectangle display list" 54 stored in memory. The subrectangle display list 54 is a set of instructions which indicate which windows are to be displayed, the size, shape and location of each window, and the relative foreground/background positions of overlapping windows. Initially, the screen process 52 adjusts the subrectangle list so that only the log in window is displayed. The contents of the log in window are controlled by "display lists" generated by the log in virtual terminal 50 in response to information transmitted to it from the UNIX operating system via device driver 44. The display lists generated by the virtual terminal are transmitted to a display list processor 56 which generates display control data for storage in the frame buffer 26 of FIG. 1. The display list processor 56 determines which windows are to be displayed, along with their size, shape and screen locations from the information contained in the subrectangle display list 54 maintained by the screen process and determines what is to appear in each displayed window or window portion from display lists maintained by the associated virtual terminal.

Each window can display either text or graphics superimposed on one another to produce the window image. When the log in virtual terminal 50 receives data from the UNIX system 40 indicating that a log in prompt is to be displayed in the log in window, it sends three display lists to the display list processor 56. The first display list tells the display list processor 56 to make the window blank by clearing both surfaces. The second display list indicates the text to be displayed and the third display list tells the list processor the graphics to be displayed. Typically, for the log in window no graphics are displayed. Since the screen process 52 has initially set the subrectangle display list 54 to indicate that the entire log in window is to be displayed, the display list processor creates and fills the entire window.

The operator selects the log in window by moving the cursor into the window and pressing a button on the mouse. The display operating system senses this action and subsequently transmits any input from the keyboard to the log in virtual terminal. As the operator enters the log in information, the data is transmitted to the log in virtual terminal 50 which prepares new display lists which blanks the text screen in the window and then writes in the log in characters typed by the operator. The virtual terminal 50 also transmits the log in information to the UNIX system which creates a new shell process for the user.

At this point the operator may collapse the log in window. To do so the operator selects the command pop-up window as described hereinabove. Pop-up windows are controlled by a pop-up process 58, also established by the display operating system during the booting operation. When the operator selects the pop-up window, the display operating system sends the X,Y coordinates of the cursor, and a signal indicating the operator has depressed the appropriate mouse button, to the screen process 52. The screen process 52 then modifies the subrectangle display list 54 to tell the display list processor 56 to display the pop-up menu window in the location indicated by the X,Y coordinates of the cursor. The screen process 52 also transmits a redraw command to the pop-up process 58 telling it to transmit the appropriate

8

display lists to the display list processor 56. The pop-up process 58 acquires the display lists associated with either the command or convenience windows from memory, the lists having been created during system boot. When the command window is displayed, and the operator selects a command, the display operating system 48 again sends the X,Y coordinates of the cursor to the screen process 52 which determines therefrom which command was selected. The screen process 52 then sends a message to the pop-up process 58 indicating the command selected and also modifies the subrectangle display list 54 so that the display list processor 56 collapses the command window. When the pop-up process 58 receives the command indication from the screen process, it calls a subroutine which performs the command.

As described hereinabove, the operator can initiate a new UNIX process by selecting the create command in the pop-up window. Each time the create command is selected, the display operating system 48 creates a new application virtual terminal 60 process. Although only one application virtual terminal 60 is shown in FIG. 4, one such virtual terminal 60 is created for each active process. The screen process 52 modifies the subrectangle display list 54 to establish the presence of the window and also transmits information to the display system server 46 to request a new process and to inform the server of the I/O socket through which the new virtual terminal may be accessed. The server 46 then requests the UNIX operation system to fork a new process for the user and establishes the path connecting the device driver 44 to the virtual terminal 60.

Whenever a window displayed on the screen is created, destroyed, collapsed, moved or resized, the screen process 52 alters the subrectangle list to effectuate the change in that window. It also alters the subrectangle list to change any other window affected by the change. For instance, when a new window is created, it may cover portions of other windows. Therefore the screen process alters the subrectangle list 54 so that the display list processor 56 knows to display only the portions of those windows not covered by the new window. The screen process 52 also sends a redraw command to every virtual terminal whose window display is affected by the new window, telling each such virtual terminal to transmit new display lists to the display list processor 56 so that the display list processor will know what to put in the displayed portions the windows. Whenever a virtual terminal modifies a display list in response to data from the associated UNIX process, it not only transmits the new display lists to the display list processor 56, it also maintains the display lists in memory so that it can retransmit them to the display list processor when it receives a redraw command from the screen process 52.

The display system 10 of the present invention, as depicted in FIG. 4, thus permits a plurality of independent processes, running in a multiprocess host computer 12, to independently control windows on the same screen. Each virtual terminal 50 or 60 remains available to receive display data from the associated process regardless of the state of operation of any other process. The display list processor 56 is adapted to update the windows as fast as the independently operating virtual terminals can produce revised display lists. From the operator's viewpoint each window is active and many windows may appear to change simultaneously. There is no need for the operator to terminate one process in

9

order to input or output access another process. Also, since each virtual terminal stores the updated display lists, there is no need to transfer display data from the frame buffer to another memory when a portion of a window is covered, or when a window is collapsed because the window display can be restored by recalling the display list. Finally, a process may remain output active even if its corresponding window is not displayed since it is only necessary that the associated virtual terminal 60 update and store the associated display list. Thus the output stream from each host computer process is maintained regardless of the state of the display.

The virtual terminals 50 and 60, the pop-up process 58 and the screen process 52 are controlled by software based state machines as illustrated by a flowchart depicted in FIG. 5. The state machines can accept and respond to up to seven input event signals, numbered 1 to 7. The state machines start in block 70 when the process is initialized. Thereafter the process moves to block 71. If a signal indicates that an event 1 has not occurred block 71 directs flow to block 72. If an event 2 has not occurred, block 72 directs the program to block 73. In a similar fashion, decision blocks 73-77 check to see if events 3-7, respectively, have occurred and if not, program flow is directed to the next decision block. If none of the events have occurred, block 77 returns operation to block 71. Whenever a decision block 71-77 detects that the corresponding event has occurred, then blocks 71-77 direct flow to corresponding action blocks 81-87, respectively. Each action block 81-87 calls a corresponding subroutine labeled action 1-7. The subroutine performs a selected action and then returns to block 71. Thus it is seen that actions 1-7 are taken in response to events 1-7, and when an action is completed the process always returns to block 71. This arrangement gives action 1 the highest priority and action 7 the lowest priority.

The event 1 input for each virtual terminal state machine is a termination signal from the display operating system indicating that the UNIX process being served by the virtual terminal is to be terminated. Action 1 therefore comprises the steps of freeing the portion of memory currently used by the virtual terminal for storing its display lists, sending a process termination message to the UNIX system and then returning an acknowledgment to the display operating system so that the operating system can destroy the virtual terminal. Event 2 for each virtual terminal is the redraw request from the screen process. In action 2 the virtual terminal performs the following steps:

1. Build a clear screen display list;
2. Submit the clear screen display list to the display list processor;
3. Wait for a display change completion message from the display list processor;
4. Build a graphics display list from data in memory;
5. Submit the graphics display list to the display list processor;
6. Wait for another completion message from the display list processor;
7. Build a text display list from data in memory;
8. Submit the text display list to the display list processor;
9. Wait for another completion message from the display list processor;
10. Return a completion message to the screen process; and

4,761,642

10

11. Exit.

The virtual terminal state machine recognizes no event 3. Event 4 is the completion message from the display processor. Action 2 is actually suspended in steps 3, 6 and 9 and the program continues to cycle through blocks 71-77 until the completion message from the display process diverts the procedure to block 85 which simply sets a redraw flag and exits. On the next pass through block 73 the program is diverted again to block 83 where action 3 is resumed.

Event 5 is a message from the device driver indicating that it wants to send data to the virtual terminal. Action 5 comprises the following steps:

1. Receive the data from the display driver and acknowledge receipt;
2. Parse the data;
3. Build a clear screen display list;
4. Submit the clear screen display list to the display processor;
5. Wait for a completion message from the display list processor;
6. Build a graphics display list from data in memory and from the UNIX process;
7. Submit the graphics display list to the display list processor;
8. Wait for another completion message from the display list processor;
9. Build a text display list from data in memory and from the UNIX process;
10. Submit the text display list to the display list processor;
11. Wait for another completion message from the display list processor; and
12. Exit.

Event 6 is an acknowledgment signal from the device driver indicating that the UNIX process has received a data packet from the virtual terminal. In action 6 the virtual terminal destroys the data packet. Event 7 is a signal from the display operating system indicating that the virtual terminal is to receive keyboard input. In action 7 the virtual terminal acquires the keyboard data, builds a data packet for transmission to the device driver, and sends the data packet to the device driver. The virtual terminal retains a copy of the data packet until it receives the acknowledgment of receipt from the device driver (event 6).

For the screen process there are no events or actions 1 or 5. Screen process event 2 is an acknowledgement from the display system server that the UNIX operating system has established a new shell for the user. In action 2, the screen process modifies the subrectangle list so that the log in window is displayed. Screen process event 3 is the acknowledgment received from a virtual terminal after the terminal has responded to a redraw command. In action 3 the screen process sends the acknowledgment to a subroutine waiting for it. Event 4 is a signal from the display list processor indicating that it has processed a background display list which controls the screen background. This background display list is maintained by the screen process and is sent to the display list processor on system start up and whenever the operator makes a change to the background color using the attribute command in the convenience menu. In action 4 the screen process forwards the acknowledgment to the subroutine. Event 6 is a request from the display operating system to create a new shell. This occurs on system boot. In action 6, the screen process

4,761,642

11

transmits the new shell message to the display system server.

Event 7 is an indication from the display operating system that the operator has moved the mouse out of the current input active window and has pressed a button. As long as the mouse is within the current input active window, the mouse input is sent to the virtual terminal behind the window and the screen process is not informed of mouse activity. Action 7 of FIG. 5 is illustrated by the flowchart of FIG. 6. Starting in block 99, the program proceeds to block 100 which passes program flow to block 101 if the first mouse key was depressed. If the cursor is over background space and not over a window or an icon, then the process moves to block 102 wherein the current input active window is input deactivated. Action 7 is then completed in block 108. If the cursor is over a window or over a collapsed window icon, block 101 directs flow to block 103 where the current input active window is input deactivated and the selected window is input activated. The action is then terminated in block 108.

If the second mouse key was depressed, the program passes from block 100 through block 104 to block 105 where a convenience menu subroutine is called and executed. If the third mouse button key was pressed, the program proceeds from block 100 through blocks 104 and 106 to a block 107 where a command menu subroutine is called and executed. If no key was pressed, or on completion of blocks 105 and 107, action 7 ends in block 108.

FIG. 7 is a flowchart detailing the convenience menu subroutine of block 105. Starting in block 110, the subroutine modifies the subrectangle list so that the display list processor can display the convenience window. Then in block 112 the screen process transmits a redraw signal to the pop-up process indicating that it should transmit the display list for the convenience window to the display list processor. The screen process also transmits the redraw command to each virtual terminal controlling a window covered by the convenience window so that these terminals also transmit new display lists to the display list process. Then in block 113 the screen process waits for the display process completion signals from the pop-up process and each affected virtual terminal. On receipt of all completion signals, then, in block 114, the screen process waits until it receives a message from the display operating system that the operator has released the selection button. In block 115 the screen process acquires the X,Y coordinates of the mouse, at the time the mouse button is released, from the display operating system and determines what command was selected. Then in block 116, the screen process again modifies the subrectangle list to collapse the convenience window, and, in block 117, transmits the redraw signal to all virtual terminals corresponding to windows uncovered when the command window collapses. In block 118 the screen process waits until it receives the display process completion messages from each affected virtual terminal.

Next, decision blocks 121-126, connected in sequence, divert the program to action blocks 131-136 respectively, if the operator has selected the redraw, block, log in, hardcopy, softcopy or set attribute commands. If no command is selected, or on completion of any action block 131-136, the subroutine ends in block 127. In block 131 the screen process changes the subrectangle list so that every window is expanded and sends a redraw command to each virtual terminal so that the

12

screen is completely redrawn. In block 132, the screen process acquires a code message from the operator and then notifies the display operating system that the display list outputs of the virtual terminals to the display list processor are to be inhibited until further notice. If such output was already inhibited, the block command causes the screen process to wait for the same code message from the operator and then to send a message to the operating system unblocking the screen. (The operator can then update the screen using the redraw command discussed hereinabove.) In block 133, the screen process requests the display server to initiate the new UNIX shell, permitting the operator to log in. In blocks 134 and 135, the screen process sends a message to the operating system requesting that the current screen should be printed or saved in memory. In block 136 the screen process modifies the subrectangle list to reflect changes in display attributes keyed in by the operator.

The detailed operation of action block 107 of FIG. 6 is flowcharted in FIG. 8. The action begins in block 139, and in block 140 the screen process determines the command selected by the operator. Block 140 includes steps substantially the same as blocks 111 to 118 of FIG. 7. Next, in decision blocks 141-148, connected in sequence, program flow is diverted to blocks 151-158, respectively, if the operator has selected the create, destroy, reframe, move, collapse, expand, bury or uncover commands. If none of these commands were selected, the action terminates in block 160. Once any action block 151-158 is completed, program flow also returns to block 160.

If the create command was selected, then in block 151, the screen process creates a new window, first by acquiring the X,Y coordinates of the upper left hand and lower right hand window corners transmitted from the display operating system in response to mouse push-button operation. The screen process then sends a message to the display system server requesting a new UNIX process, and waits for a reply from the display system server. When the server replies, the screen process requests the display operating system to create a new virtual terminal. The screen process then modifies the subrectangle list, sends a redraw command to all affected virtual terminals, and waits for a reply before completing the action block.

When the operator selects the destroy command, program flow is directed to block 152 wherein the screen process modifies the subrectangle list to eliminate the window to be destroyed and sends a terminate message to the corresponding virtual terminal. It also sends a draw message to any virtual terminal controlling a window being uncovered. If the reframe command is selected, then in block 153 the screen process acquires the upper left hand and lower right hand window coordinates from the display operating system in response to cursor movement and mouse button operation, changes the subrectangle list and sends a redraw message to all affected windows.

If the move command was selected, then in block 154 the screen process acquires the new screen X,Y coordinates for the upper left hand corner of the window being moved, changes the subrectangle list to effectuate the move, and sends a redraw message to the virtual terminals behind all affected windows. If the collapse command is selected, then in block 155 the screen process changes the subrectangle list to remove the window and sends a redraw command to the virtual termi-

4,761,642

13

nals controlling every uncovered window. If the expand command is selected, then in block 156 the subrectangle list is modified so that the selected window is displayed and a redraw message is sent to its virtual terminal and to all other virtual terminals behind windows being covered by the expanded window. If the bury command is selected, then in block 157 the screen process changes the subrectangle list to put the window behind any overlapping windows and sends the redraw command to all affected virtual terminals. Finally, if the uncover command is selected, the screen process changes the subrectangle list to put the selected window on top of all overlapping windows and sends the redraw command to each affected virtual terminal.

Thus the display system of the present invention permits multiple active processes to simultaneously display and update their outputs on a single screen and permits an operator to quickly input access any one of the processes at any time. Further, the window operations performed by the system are transparent to the host process applications since each application is permitted independent access to its own virtual terminal.

While a preferred embodiment of the present invention has been shown and described, it will be apparent to those skilled in the art that many changes and modifications may be made without departing from the invention in its broader aspects. The appended claims are therefore intended to cover all such changes and modifications as fall within the true spirit and scope of the invention.

I claim:

1. In a computer system having a memory and having processing means for concurrently executing multiple processes, including first processes, each of which first

14

processes receives input data produced by a computer terminal of said computer system and produces output data for controlling display on a screen of said computer terminal, a method for permitting concurrent data communication between a plurality of said first processes and a single computer terminal, the method comprising the steps of:

initiating and concurrently executing for each of said first processes a corresponding virtual terminal process each virtual terminal process receiving input data from the single computer terminal and forwarding said input data to the corresponding first process, receiving output data produced by the corresponding first process, and maintaining a separate display list in said memory, said display list comprising data defining a display in accordance with said output data produced by the corresponding first process;

initiating and executing a screen process for maintaining a subrectangle list in said memory, said subrectangle list comprising a set of instructions defining positions and sizes of display windows to be displayed on a screen of said single computer terminal, each window corresponding with a separate display list; and

initiating and executing a display list process for periodically transmitting display data to said single computer terminal for causing said single computer terminal to concurrently display said display windows on said screen at positions and of sizes defined by said subrectangle list, each window including a display in accordance with the display defined by the corresponding subrectangle list.

* * * * *